Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 72 (2015) 186 – 193

Procedia Computer Science

The Third Information Systems International Conference

Improving Automated Programming Assessments: User Experience Evaluation Using *FaSt-generator*

Rohaida Romli[a], Shahida Sulaiman[b], Kamal Zuhairi Zamli[c]

[a] *School of Computing(SOC), College of Arts and Sciences, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia*
[b] *Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia*
[c] *Faculty of Computer System and Engineering, Universiti Malaysia Pahang, Lebuhraya Tun Razak, 26300 Gambang, Kuantan, Pahang, Malaysia*

Abstract

Automatic Programming Assessment (APA) has been known as a method used to automatically mark and grade students' programming solutions. In order to realise APA as a tangible deliverable, a number of automated tools which are called Automated Programming Assessment Systems (APAS) have been developed and tested for decades. Basically, the need for decreasing the load of work among lecturers, timely feedback to students and accuracy on the grading results are the common reasons that motivate the need for APAS. In order to carry out a dynamic testing in APA, it is necessary to prepare an appropriate and adequate set of test data to judge the correctness quality of students' programming solutions in terms of the aspects of functional and/or structural testing. Manual preparation of quality test data becomes a hard, time consuming, and feasible task in the practice of both software testing and APA. Thus, the generation of automated test data is highly desirable to alleviate the humans' burden from performing repetitive tasks. This paper aims to describe the design, implementation and user experience when evaluating a tool developed to support APA as a test data generator that is called *FaSt-generator*. The tool plays an important role to furnish a test set that includes an adequate set of test data to execute both the functional and structural testing in APA. Results collected from the conducted user experience evaluation using *FaSt-generator* reveal that all the subjects had relatively positive opinions and greatly favour the criteria of User Perception and End-User Computing Satisfaction (EUCS).

Peer-review under responsibility of organizing committee of Information Systems International Conference (ISICO2015)

Keywords: automatic programming assessment; test data generation; functional testing; structural testing; user experience

* Corresponding author. Tel.: +6-04-928-5188; fax: +6-04-928-5067.
E-mail address:aida@uum.edu.my.

Peer-review under responsibility of organizing committee of Information Systems International Conference (ISICO2015)
doi:10.1016/j.procs.2015.12.120

1. Introduction

Automatic Programming Assessment (APA) has recently become a significant method in assisting educators to automatically assess and grade students' programming exercises. By applying this method, the educators' workload can be reduced effectively since the typical manual assessments are always prone to errors and lead to inconsistencies. Practically, APA offers important benefits in terms of immediate feedback, objectivity and consistency of the evaluation as well as substantial savings of time in the evaluation of the assignments [1] without the need to reduce exercises [2]. Besides, APA improves consistency, accuracy and efficiency of assessments [3].

To date, with the intention of realising APA as a physical deliverable, a lot of automated tools called Automated Programming Assessment Systems (APAS) have been developed and tested. Among these tools are Assyst [3], BOSS [4], GAME [5], TRAKLA2 [6], PASS [7], ELP [8], CourseMaster [9], WeBWorK-JAG [10], SAC [11], Oto [12], ICAS [13], PETCHA [14], eGrader [15], and Bottlenose [16]. This large pool of tools depicts that such systems provide advantages not only to lecturers, but might also play an important role in students' learning outcomes [17]. Albeit the availability of diverse automated tools, researches with regard to APA continue to mature. Most of the recent studies keep on improving the features of existing APAS and employing different strategies in producing more accurate markers as well as providing better functions. APAS mainly aims to provide a more accurate assessment that is the key to assess students in achieving the learning outcomes of the programming course.

By default, APA requires a test data generation process to perform a dynamic testing. Thus far, various automated methods for test data generation are accessible particularly in the software testing field, yet they are rarely utilized in recent studies of APA. There have been several early attempts to integrate APA and test data generation, but most of the studies usefully execute tests and evaluate test results automatically; much of which have not sufficiently and systematically automated the generation of adequate test data and test set besides attempting to exclude the use of particular lecturers' knowledge in test cases design [6][18][19][20]. In addressing this gap, this study proposes a framework of test data generation to derive the desired test data and test set to perform both of the functional and structural testing for APA. Our previous works [21][22][23] provide the details of the proposed framework. In realising the framework, a tool named *FaSt-generator* was developed to furnish a tangible deliverable and to provide a systematic and consistent way of deriving and generating test data among different individual lecturers. This is regardless of whether or not the lecturers have an optimal expertise in the knowledge of test cases design. This paper aims to investigate user experience evaluation when using the developed tool.

This paper consists of the three consecutive sections after this Introduction section. Section 2 provides a description of design and implementation of *FaSt-generator*. Section 3 reveals the analysis and findings from the conducted user experience evaluation using *FaSt-generator* in terms of the criteria of User Perception and End-User Computing Satisfaction (EUCS). Finally, Section 4 concludes the paper.

2. Design and Implementation of *FaSt-generator*

FaSt-generator is a module that involves one actor that is a lecturer. The lecturer is a person who teaches Java programming course and plays an important role in preparing and managing the sources needed in executing the assessment process. Such sources are topics of a course syllabus, notes and programming exercises for each topic, program schema (in solution model), the generated schema of test set which includes its corresponding test data and input condition and, weight and score values for measuring the correctness of students' program.

The role that lecturers play in using *FaSt-generator* reflects the function that they can invoke to manipulate sources needed in implementing the assessment process. *FaSt-generator* consists of nine use cases as depicted in Fig. 1. Each of the use case represents a function that a lecturer can invoke in *FaSt-generator*. The <<*include*>> stereotype indicates that each invocation of a particular use case will invoke the included use case at least once [24]. For example, before the lecturers invoke any use cases or functions, they must be the valid users. Thus, they must successfully log on into *FaSt-generator*. The

<<*extend*>> stereotype indicates that one use case may be extended by another and it is used to identify when a use case can optionally be extended by functionality in another use case [25]. As shown in the figure, every time the lecturers want to manage assignments, they can create new assignments or edit assignments. Also, every time the lecturers desire to create a new assignment or edit the existing assignment, they can prepare a solution model or prepare a test data. Interfaces of *FaSt-generator* were designed based on the use cases as defined in Fig.1. This paper only reveals the most crucial test case that is Prepare Test Data. Prepare Test Data use case concerns the part that generates the desired test data and test set schema. This section only presents the sample of interfaces for one category of structural test data generation. Our previous works [22][23] provide the details of the categories. Fig. 2 presents the sequence of interfaces representing steps taken to generate test set and test data for the category of selection control structure that falls under structural test data generation. It needs five main steps to complete the process of generating the desired test set and test data.

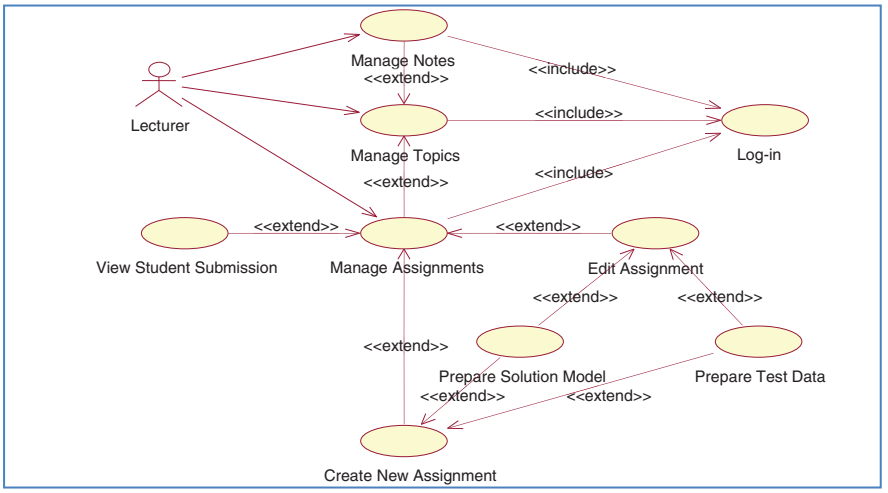
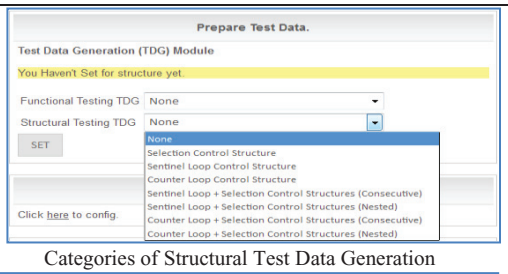
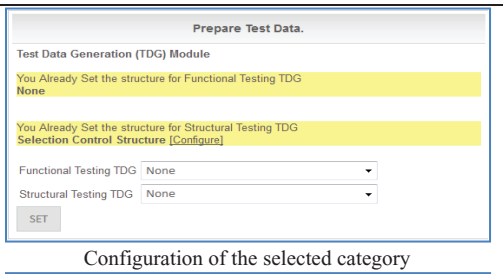


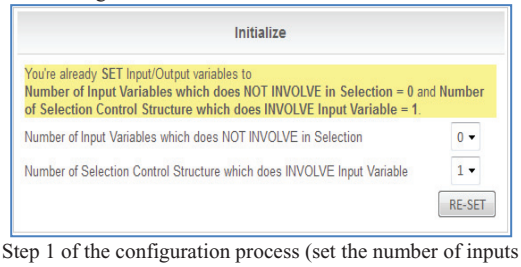
Fig. 1. Use case diagram of *FaSt-generator*



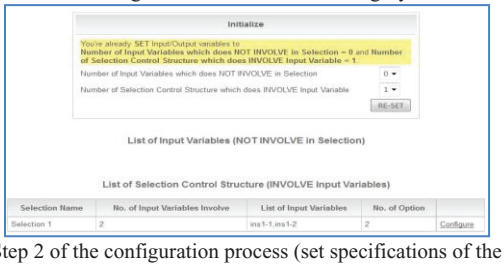
Categories of Structural Test Data Generation



Configuration of the selected category



Step 1 of the configuration process (set the number of inputs outside control structure and the number of selection control structures involved)



Step 2 of the configuration process (set specifications of the identified selection control structure)

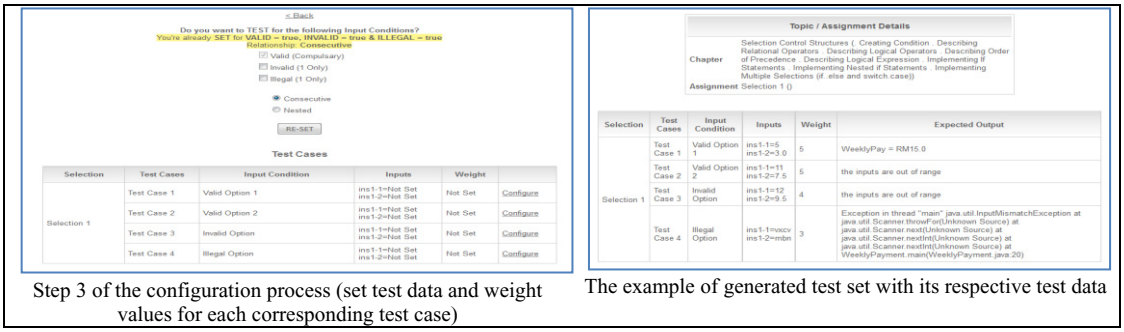


Fig. 2. Sequence of interfaces to prepare test data for the category of selection control structure of structural test data generation

3. Results and Discussion

This section presents the analysis and findings for the user experience evaluation against *FaSt-generator* on the criteria of User Perception towards its usage (perceived usefulness, output quality, task relevance, attitude towards use, and behavioural intention to use), and EUCS (content, accuracy, and overall satisfaction). This evaluation was part of a controlled experiment conducted among lecturers who have been teaching an introductory programming course in one of the public universities in Malaysia. There were 12 subjects involved in this evaluation. A questionnaire was used as the instrument to collect the related data. In terms of the demography of subjects, the major items include: level of appointment, experience in teaching programming courses, the current way of marking students' programming exercises, and the way of preparing test data (or inputs) in programming assessment. Fig. 3 shows the frequency of responses for each item.

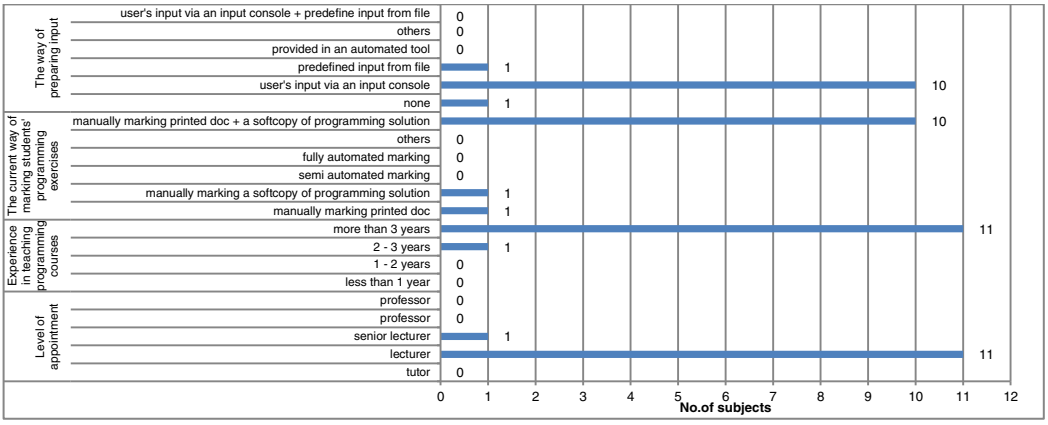


Fig. 3. Demography of the subjects

As shown in the figure, all the subjects were lecturers and a senior lecturer with their respective percentage of 91.7% and 8.3%. All of them are categorised as experienced lecturers as they have more than two years (which is equivalent to four semesters) of experience in teaching programming courses. The figure also shows that programming courses have gained much interest among junior lecturers as compared to lecturers with more experience. Focusing on the current means of marking students' programming exercises, the findings also depict that more than 80% of the lecturers did the marking manually. In terms of the way of preparing inputs or test data to assess the quality of program correctness, about 83% of the lecturers prepared them based on users' input via an input console of Java editor such as JGrasp, Eclipse, or NetBeans while executing students' programming solutions. There were 8% of them used a predefined file such as a text file to store the required test data and execute students' program

against the file via a command prompt. Only one lecturer did not execute dynamic testing to assess students' program. This shows that the lecturer performed static analysis (walkthrough) manually to assess the program. In the following sub-sections, the findings on the criteria of User Perception and EUCS are presented.

3.1 User Perception

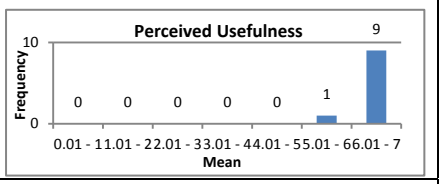
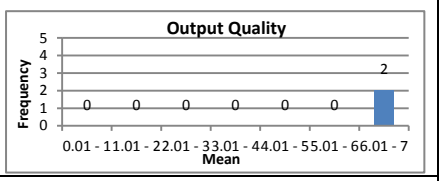
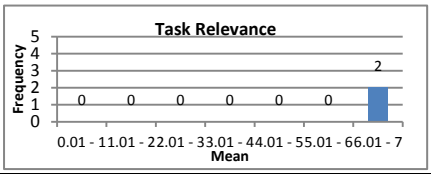
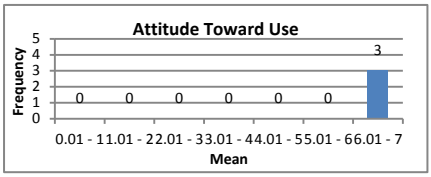
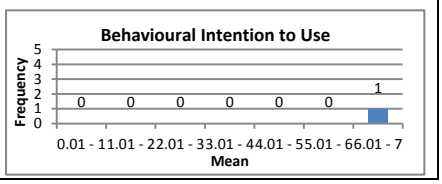
The criteria of User Perception consist of five constructs. The criteria are adapted from TAM [26] and UTAUT [27]. The five constructs include, Perceived Usefulness (PU), Output Quality (OQ), Task Relevance (TR), Attitude Towards Use (ATU) and Behavioural Intention to Use (BI) with ten, two, two, three and one items respectively in each construct. All items in each construct were measured on a 7-point Likert scale [28], where *1 = strongly disagree*, *2 = moderately disagree*, *3 = somewhat disagree*, *4 = neutral (neither disagree nor agree)*, *5 = somewhat agree*, *6 = moderately agree*, and *7 = strongly agree*. Table 1 tabulates all the constructs with their respective items. The analysis and findings of each construct in the form of descriptive statistic is tabulated in Table 2.

Table 1. Constructs and items of User Perception criteria

B1) PERCEIVE USEFULNESS (PU)	
B1-1	Using FaSt-generator improves the quality of preparing an adequate set of test data for APA.
B1-2	Using FaSt-generator gives me greater control to perform APA.
B1-3	FaSt-generator enables me to prepare an adequate set of test data for APA more quickly.
B1-4	FaSt-generator supports critical aspects of preparing an adequate set of test data for APA.
B1-5	Using FaSt-generator increases the productivity of APA.
B1-6	Using FaSt-generator improves the performance of APA.
B1-7	FaSt-generator allows me to prepare an adequate set of test data than would otherwise is possible.
B1-8	Using FaSt-generator enhances the effectiveness of APA.
B1-9	Using FaSt-generator makes it easier to prepare an adequate set of test data for APA.
B1-10	Overall, I find that FaSt-generator is useful for APA.
B2) OUTPUT QUALITY (OQ)	
B2-1	The quality of the generated test data that I get from FaSt-generator is high.
B2-2	I have no problem with the quality of FaSt-generator output.
B3) TASK RELEVANCE (TR)	
B3-1	In performing programming assessment, usage of FaSt-generator is important.
B3-2	In performing programming assessment, usage of FaSt-generator is relevant.
B4) ATTITUDE TOWARD USE (ATU)	
B4-1	I like the idea of using FaSt-generator.
B4-2	I have generally favourable attitude in using FaSt-generator.
B4-3	I believe it is a good idea to use FaSt-generator to support APA.
B5) BEHAVIOURAL INTENTION TO USE (BI)	
B5-1	If I have access to use FaSt-generator, I intend to use it to prepare test data in assessing programming exercises.

As shown in the table, the subjects are considered agreeing with each of the construct if the mean value is greater or equal than 5.0 (the scale *somewhat agree*). It is shown that for all constructs, all subjects scored means of between 5.01 and 6, and 6.01 and 7. This indicates that 100% of the subjects agreed on all the items in all the constructs. In terms of the construct of PU, it can be concluded that the greater the mean value reveals that *FaSt-generator* would enhance the performance of preparing an adequate set of test data for APA more. As for the construct of OQ it indicates the degree to which the test data derived from *FaSt-generator* matches the required criteria of test data adequacy (an integration of positive and testing criteria). Referring to the construct of OQ, the results conclude that *FaSt-generator* is relevant and applicable to perform APA. Apart from PU and OQ findings, the conclusion that can be derived from the construct of ATU is that all the lecturers affectively showed a positive reaction towards *FaSt-generator*. Finally, for the construct of BI, it can be summarised that all the lecturers seem to have a high interest to use the tool if it is available.

Table 2. Analysis and findings of the criteria User Perception

Construct	Cumulative Mean - Descriptive Statistics	Graphing Frequency of Means
Perceived Usefulness (PU)	N: MeanPU 10, Valid N (listwise) 10; Min 5.83; Max 6.58; Mean 6.3400; Std. Deviation .22823	
Output Quality (OQ)	N: MeanOQ 2, Valid N (listwise) 2; Min 6.25; Max 6.42; Mean 6.3350; Std. Deviation .12021	
Task Relevance (TR)	N: MeanTR 2, Valid N (listwise) 2; Min 6.25; Max 6.25; Mean 6.2500; Std. Deviation .00000	
Attitude Toward Use (ATU)	N: MeanATU 3, Valid N (listwise) 3; Min 6.75; Max 6.83; Mean 6.8033; Std. Deviation .04619	
Behavioural Intention to Use (BI)	N: MeanBI 1, Valid N (listwise) 1; Min 6.42; Max 6.42; Mean 6.42; Std. Deviation .	

3.2 End-User Computing Satisfaction (EUCS)

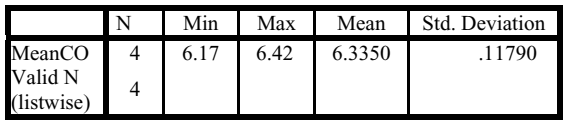
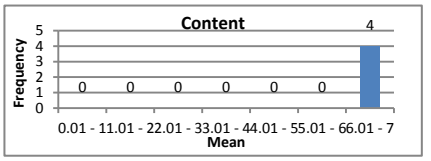
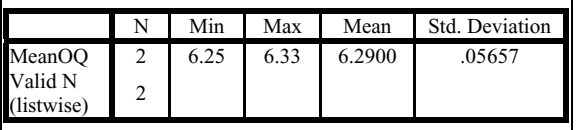
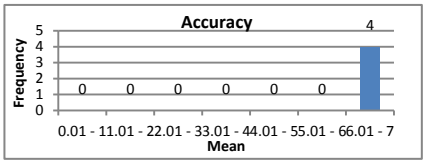
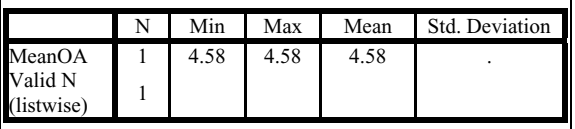
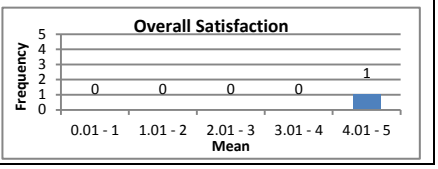
The criteria of EUCS are adapted from Doll and Torkzadeh [29]. It consists of three constructs, which are Content (CO), Accuracy (AC), and Overall Satisfaction (OA). Table 3 depicts the related constructs and items. All items in the constructs were measured based on the 7-point Likert and a 5-point scale (only for the construct of OA), similar to the criterion of User Perception. Thus, the result of each construct of EUCS in the form of descriptive statistic is tabulated in Table 4.

Table 3. Constructs and items of EUCS criteria

C1) CONTENT (CO)	
C1-1	FaSt-generator does provide the precise criteria to generate test data for APA that I need.
C1-2	The criteria use to generate test data for APA meets my need.
C1-3	FaSt-generator does provide the schema of test set that seem to be just about exactly what I need.
C1-4	FaSt-generator does provide sufficient criteria in generating test data for APA.
C2) ACCURACY (AC)	
C2-1	FaSt-generator is accurate.
C2-2	I am satisfied with the accuracy of FaSt-generator.
C2-3	FaSt-generator is accurate.
C3) OVERALL SATISFACTION (OA)	
C3-1	Overall, how would you rate your satisfaction with FaSt-generator?

As depicted in the table, similar to the criteria of User Perception, all of the subjects are considered agreed with each of the construct if its mean value is greater than 5.0 (the scale *somewhat agree*). Considering the construct of CO, it appears that all lecturers had a very positive attitude towards the content of *FaSt-generator* particularly in terms of the criteria used to generate test data. Also, in terms of the construct of AC, it can be concluded that the lecturers had affective attitude towards the accuracy of *FaSt-generator*. While, for the construct of OA, it can be deduced that the lecturers were satisfied with the overall function of *FaSt-generator*.

Table 4. Analysis and findings of the criteria EUCS

Construct	Cumulative Mean - Descriptive Statistics	Graphing Frequency of Means
Content (CO)		
Accuracy (AC)		
Overall Satisfaction (OA)		

4. Conclusion

This paper has described the design, implementation and user experience evaluation of *FaSt-generator*, a tool to automatically generate and derive the desired test data and test set to execute both the functional and structural testing for APA. This tool plays an important role to support APA as a test data generator in order to contribute as a means to educators of programming courses. It allows these educators to furnish an adequate set of test data systematically regardless of having the optimal expertise in the particular knowledge of test cases design. A systematic way to automatically derive and generate an adequate set of test data for APA would generally be useful and beneficial to lecturers of programming courses. Besides, the effort to improve the shortcomings can enhance the features of APAS especially for the part of feedbacks received by students' as the final assessment result. Furthermore, sufficient feedback is an integral part in APA as it allows students to develop their programming skill through learning from unexpected mistakes made by them. The feedback can also improve the means of assessing the quality of students' programs so as to make the assessment more accurate. In *FaSt-generator*, besides the function of deriving and generating an adequate test data, lecturers can adjust the way to allocate a portion of marks to both functional and structural testing. This feature allows a diverse way of allocating marks in marking students' programming exercise, which may depend on the needs of individual lecturers and differences between programming exercises.

Moreover, the findings from the conducted experiment to evaluate users' experience when using *FaSt-generator* in terms of the criteria User Perception and EUCS imply that most of the subjects of the experiment had relatively positive opinions with regard to all constructs of the evaluation criteria. Thus, this study anticipates the growth of this kind of tool to support lecturers in managing the process of generating an adequate set of test data in more systematic ways. In the nutshell, *FaSt-generator* is an alternative to support APAS mainly in Malaysia as limited universities have benefited from such tools.

References

[1] Aleman JLF. Automated Assessment in Programming Tools Course. *IEEE Transactions on Education* 2011; **54(4)**: 576-581.

[2] Ihantola P, Ahoniemi T, and Karavirta V. Review of Recent Systems for Automatic Assessment of Programming Assignments. In Proceedings of the 10th Koli Calling International Conference on Computing Education Research; 2010, p. 86-93.

[3] Jackson D and Usher M. Grading Student Programs using ASSYST. In Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education; 1997, p. 35–339.

[4] Luck M and Joy MS. Secure On-line Submission System. *Journal of Software – Practise and Experience* 1999; **29(8)**: 721-740.

[5] Blumenstein M, Green S, Nguyen A and Muthukkumarasamy V. GAME: A Generic Automated Marking Environment for Programming Assessment. In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04); 2004, Vol. 2, p. 212-216.

[6] Malmi L, Karavirta V, Korhonen A, Nikander J, Seppala O and Silvasti P. Visual Algorithm Simulation Exercise System with Automatic Assessment: TRAKLA2. *Informatics in Education* 2004; **3(2)**: 267-288.

[7] Choy M, Nazir U, Poon CK and Yu YT. Experiences in Using an Automated System for Improving Students' of Computer Programming. Lecture Notes in Computer Science Learning. Springer Berlin/ Heidelberg; 2005, p. 267 – 272.

[8] Truong N, Bancroft P and Roe P. Learning to Program Through the Web. *ACM SIGCSE Bulletin* 2005; **37(3)**: 9-13.

[9] Higgins CA, Gray G, Symeonidis P, and Tsintsifas A. Automated Assessment and Experiences of Teaching Programming. *Journal of Educational Resources in Computing* 2006; **5(3)**: Article 5.

[10] Gotel O, Scharff C and Wildenberg A. Extending and Contributing to an Open Source Web-Based System for the Assessment of Programming Problems. In Proceedings of the 5th International Symposium on Principles and Practice of Programming in Java (PPPJ'07), Lisboa, Portugal; 2007, p. 3-12.

[11] Auffarth B, Lopez-Sanchez M, Miralles JC, and Puig A. System for Automated Assistance in Correction of Programming Exercises (SAC). In Proceedings of the fifth CIDUI – V International Congress of University Teaching and Innovation; 2008.

[12] Tremblay G, Gu´erin F, Pons A, and Salah A. Oto, A Generic and Extensible Tool for Marking Programming Assignments. *Journal of Software- Practice and Experience* 2008; **38 (3)**: 307-333

[13] Nunome A, Hirata H, Fukuzawa M and Shibayama K. Development of an E-learning Back-end System for Code Assessment in Elementary Programming Practice. In Proceeding of the 38th Annual Fall Conference on SIGUCCS, Norfolk, VA, USA; 2010, p. 181-186.

[14] Queiros R and Leal JS. PETCHA- A Programming Exercises Teaching Assistant. In Proceeding of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'12), Haifa, Israel; 2012, p. 192-197.

[15] Shamsi FA and Elnagar A. An Intelligent Assessment Tool for Students' Java Submissions in Introductory Programming Courses. *Journal of Intelligent Learning Systems and Applications* 2012; **4(1)**: 59-69.

[16] Sherman M, Bassil S, Lipman D, Tuck N and Martin F. Impact of Auto-Grading on an Introductory Computing Course. *Journal of Computing Sciences in Colleges* 2003; **28(6)**: 69-75.

[17] Malmi L, Saikkonen R and Korhonen A. Experiences in Automatic Assessment on Mass Courses and Issues for Designing Virtual Courses. In Proceedings of The 7th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE' 02), Aarhus Denmark; 2002, p. 55-59.

[18] Shukur Z, Romli R and Hamdan AB. Skema Penjanaan Data dan Pemberat Ujian Berasaskan Kaedah Analisis Nilai Sempadan (A Schema of Generating Test Data and Test Weight Based on Boundary Value Analysis Technique). *Technology Journal* 2005; **42(D)**: 23-40.

[19] Ihantola P. Automatic Test Data Generation for Programming Exercises with Symbolic Execution and Java PathFinder. Master Thesis of Helsinki University of Technology, Finland; 2006.

[20] Tillmann N, Halleux JD, Xie T, Gulwani S and Bishop J. Teaching and Learning Programming and Software Engineering via Interactive Gaming. In Proceedings of the 2013 International Conference on Software Engineering (ICSE'13), San Francisco,CA, USA; 2013, p. 1117-1126.

[21] Romli R, Sulaiman S, and Zamli KZ. (2014), Test Data Generation Framework for Automatic Programming Assessment, Proceeding of 8th Malaysian Software Engineering Conference 2014 (MySEC'14), Langkawi, Malaysia.

[22] Romli R, Sulaiman S, and Zamli KZ. (2013), Designing a Test Set for Structural Testing in Automatic Programming Assessment, *International Journal of Advances in Soft Computing and Its Application (Special Issues on Application of Soft Computing in Software Engineering)* 2013; **5(3)**: 41-64.

[23] Romli R, Sulaiman S, and Zamli KZ. Test Data Generation in Automatic Programming Assessment: The Design of Test Set Schema for Functional Testing. Proceeding of 2nd International Conference on Advancements in Computing Technology (ICACT'11), Jeju Island, South Korea; 2011, p. 1078-1082.

[24] Conallen J. *Building Web Applications with UML*. 2nd Edition. Pearson Education, Inc: USA; 2003.

[25] Roff JT. *UML: A Beginner's Guide*. McGraw-Hill/Osborne, California: USA; 2003.

[26] Davis FD. User Acceptance of Information Technology: System Characteristics, User Perceptions and Behavioural impacts. *International Journal of Man-Machine Studies* 1993; **38 (3)**: 475-487.

[27] Venkatesh V, Morris MG, Davis GB, and Davis FD. (2003). User acceptance of information technology: toward a unified view**.** *Journal MIS Quarterly* 2003; **27 (3)**: 425-478.

[28] Ajzen I, and Fishbein M. *Understanding Attitudes and Predicting Social Behaviour*. Prentice-Hall, Englewood Cliffs: NJ; 1980.

[29] Doll WJ, and Torkzadeh G. The Measurement of End-User Computing Satisfaction, *MIS Quarterly* 1988; **12 (2)**: 259-274.